AMENDMENTS TO THE SPECIFICATION:

Page 1, amend paragraph [0001] as:

[0001] This invention relates to an ~~a kind of~~ architecture of method for fetching microprocessor's instructions, ~~in which the method is supposed to read two instructions in program memory in the event of a conditional branch, otherwise read an instruction only~~ and more particularly to the pre-fetching of program instructions in the event of a conditional branch for reducing power consumption.

Page 1, cancel paragraph [0002].

Page 1, amend paragraph [0003] as:

[0003] The effectiveness of a computer is often evaluated based on its processing speed of an instruction. A single-cycle instruction is an instruction that can be executed and completed within a cycle and in the mean time allows ~~allow~~ a microprocessor to pre-fetch ~~pre-read~~ the next instruction~~, however, as a matter of fact, it isn't the case that all the instructions in a program are single cycle instructions~~. However, not all instructions in a program are single cycle instructions. How to reduce the processing time of an instruction has been a great concern to the designers of computer processors.

Page 1, amend paragraph [0004] as:

[0004] When executing the general logic instructions ~~shown in Fig. 1~~ of a program, a microprocessor is supposed to [[have]] run an instruction and pre-fetch ~~consecutively pre-read~~ the next one totally in an instruction cycle by adding value 1 to a program counter (PC). The next instruction can thus be executed in the next cycle. Therefore, single cycle

instructions can be executed consecutively. As shown in Fig. 1, while instruction N is executed, instruction N+1 is also pre-fetched for execution in the next cycle. However, if an instruction executed is a "CALL" instruction, the pre-fetched instruction would not be executed in the next cycle because the "CALL" instruction is supposed to jump to a different address specified in the "CALL" instruction. To accomplish this instruction jump~~If the next instruction is a "CALL" instruction~~, the PC will be added with a discrete variable "M" specified in the "CALL" instruction instead of the usual 1 to make the PC value discontinuous. Before so doing, the program would need a no operation (NOP) instruction for loading the correct address of the variable "M" to the PC for fetching and executing the instruction called by the "CALL" instruction. ~~Such a makeshift~~ The insertion of the no operation requires at least one more instruction cycle that usually deteriorates the microprocessor's effectiveness.

Page 2, amend paragraph [0005] as:

[0005] In order to overcome the inefficiency of the additional instruction cycle required, ~~As to improve abovesaid defect,~~ the procedure of an existing method for fetching instructions shown in Fig. 2 is to pre-fetch ~~pre-read~~ instructions at address N+1 and N+2 while ~~in the meanwhile~~ the instruction at address N is executed, and at this moment, the method also decodes the N+1 instruction. In case the N+1 instruction is found not a general logic instruction, such as a "CALL" instruction for example, the next instruction to be executed will be replaced by a "NOP" instruction for loading the correct address of the variable "M" specified in the "CALL" instruction and pre-fetching ~~and pre-reading~~ the instructions at address "M" and "M"+1[[,]] so that the called instruction at

address "M" will be executed in the next instruction cycle for eliminating the ~~waiving of the abovesaid~~ extra cycle to thereby improve ~~heighten~~ the processing effectiveness. As shown in Fig. 2, the prior art always fetches two additional instructions because in the case of a conditional branch instruction, the target address of a conditional branch instruction can be either one of the following two instructions. The pre-fetching of two instructions continuously while executing the program increases the consumption of power.

Page 2, amend paragraph [0006] as:

[0006] ~~Nevertheless~~ Although the efficiency is improved in the above method, more power is consumed during the process of fetching and storing those two instructions ~~previously, and it is considered still rooms available for improvement of power consumption~~ and there is a need to further improve the pre-fetching of instructions for reducing power consumption.

Page 2, amend paragraph [0007] as:

[0007] The primary object of this invention is to provide a ~~kind of architecture of~~ method for fetching microprocessor's instructions. The method which normally pre-fetches ~~pre-reads~~ a next instruction would pre-fetch ~~pre-read~~ and pre-decode two next instructions in case it encounters a conditional branch ~~"CALL"~~ instruction so as to waive unnecessary reading of program memory and reduce power consumption accordingly.

Pages 2-3, amend paragraph [0008] as:

[0008] Another object of this invention is to provide an ~~a kind of~~ architecture ~~of method~~ for practicing the method of fetching microprocessor's instructions. In the process

of executing instructions, a processing unit is employed to decode an instruction next to the current one for setting the state of an instruction reading-amount register. ~~In the case~~ If the next instruction is found a conditional branch instruction, both an odd and an even address buffer register are enabled simultaneously for fetching two next instructions, wherein the choice of an immediate one is determined by the processing unit. [[Then]] If the next instruction is not a conditional branch instruction, only one of the address buffer register is enabled for fetching an instruction in order to waive any unnecessary reading of program memory for reducing power consumption.

Page 4, amend paragraph [0012] as:

[0012] A next instruction succeeding to the current one might have several alternatives, including: an only address made by adding 1 to the present PC (program counter) value of the first category; [[or,]] a new and only address contained in the current instruction of the second category; [[or,]] a new and only address contained in the current instruction or a return address found in a stack of the third category~~, in which a return address can be found in the stack~~; or[[,]] an address at PC+1 or PC+2 of the fourth category[[,]] which is to be determined by a processing unit. Therefore, when a succeeding instruction is decoded as a conditional branch instruction, a method of this invention for fetching microprocessor's instructions ~~is supposed to pre-read and pre-decode~~ pre-fetches and pre-decodes two sequential instructions and chooses ~~choose~~ to execute one of those alternatives.

Pages 4-5, amend paragraph [0013] as:

[0013] As shown in Fig. 3, in running a program, the method of this invention shall ~~enter to~~ choose one of four options after execution of a buffer step **301** and a pre-fetching ~~pre-reading~~ step **302**. If the next instruction is decoded and found a general logic instruction **303** for example, the procedure of this method is to add 1 to [[a PC]] the PC value (namely, PC+1) **307** and set an instruction reading-amount register in a state for fetching a next instruction only **311**. If ~~; or, if~~ it is found an unconditional branch instruction **304**, the PC will point to a new address **308** and set the instruction reading-amount register in a state for fetching an instruction **311**. If ~~; or, if~~ it is found a "CALL" or a "RETURN" instruction **305**, the PC will point to a new address **309** and set the instruction reading-amount register in a state for fetching a specified instruction only **311**, or if it is found a conditional branch instruction **306**, the PC will point to a next (PC+1) and then a further next address **310** (PC+2) and set the instruction reading-amount register in a state for fetching two instructions **312** for the processing unit to choose and execute one of the alternatives **313**.

Page 5, amend paragraph [0014] as:

[0014] Fig. 4 is an ~~a kind of~~ architecture embodiment of the method of this invention for fetching microprocessor's instructions. In Fig. 4, by taking advantage of an instruction reading-amount register **411**, which is set to binary "1" for reading two instructions when a processing unit **410** has pre-fetched ~~pre-read~~ and pre-decoded a conditional branch instruction, namely, the method will read two instructions instead of one in the next instruction cycle. On the contrary, the instruction reading-amount register **411** is set to

binary "0" for reading one instruction when the processing unit **410** has pre-fetched ~~pre-read~~ the next instruction and found it in a form other than the conditional branch.

Pages 5-6, cancel paragraph [0016] and replace with the following new paragraphs:

As shown in Fig. 4, an incremental circuit **401** increments the value of the address lines. Both the address lines and the output of the incremental circuit **401** are connected to multiplexers **402, 403**. The selection switch of multiplexer **402** is connected to the least significant bit (LSB) of the address lines to control the output of the multiplexer **402**. The selection switch of multiplexer **403** is connected to the least significant bit (LSB) of the output of the incremental circuit **401** to control the output of the multiplexer **403**.

The outputs of the two multiplexers **402, 403** are sent to the even and odd address buffer registers **404, 405** respectively. Multiplexers **414, 413** control the two address buffer registers **404, 405** respectively for the fetching of even-page and odd-page program memory portions **406, 407**. A multiplexer **408** which is controlled by the multiplexer **412** selects either even-page or odd-page program memory **406, 407** output and sends it to the instruction buffer register **409** for execution in the processing unit **410**.

In a first example, the instruction reading-amount register is "0" and the address lines have a value "10". As a result, multiplexers **402** and **403** output address values "10" and "11" respectively based on their respective selection switches. The even address buffer register **404** is enabled by the multiplexer **414** to fetch the even-page program memory portion **406** and send the fetched instruction to the instruction buffer register **409** through the multiplexer **408**.

In a second example, the instruction reading-amount register is "0" and the address lines have a value "11". As a result, the multiplexer **402** chooses the incremented address value "12" while the multiplexer **403** chooses the address value "11". The odd address buffer register **405** is enabled by the multiplexer **413** to fetch the odd-page program memory portion **407** and send the fetched instruction to the instruction buffer register **409** through the multiplexer **408**.

In a third example, the instruction reading-amount register is "1" and the address lines have a value "11". In this case, the multiplexer **402** chooses the address "12" while the multiplexer **403** chooses the address "11", and both the odd and the even address buffer registers **405**, **404** are enabled to make the odd-page and the even-page program memory readable. The address chosen by the instruction buffer register **409** is determined by the processing unit **410** because the selection switch "S" of the multiplexer **412** is controlled by the processing unit **410**.

Pages 6-7, amend paragraph [0018] as:

[0018] Fig. 5 shows an example of executing a program according to the method of this invention. In processing a conditional branch instruction at PC address 10, the program pre-fetches ~~pre-reads~~ an unconditional branch instruction at PC address 11, and when processing the unconditional branch instruction, the program pre-fetches ~~pre-reads~~ an instruction at a next address. Referring to the timing chart of Fig. 5, when the instruction at address 9 is executed and [[when]] the pre-fetched ~~pre-read~~ next instruction at address 10 is decoded as a conditional branch instruction for example, the instruction reading-amount register is set to "1" ~~such that the time~~ so that two following instructions

at address 11 and 12 will be pre-fetched when the instruction at address 10 is executed~~, the instruction at address 11 and 12 are pre-read~~. If the instruction at address 11 is chosen and found by the processing unit as an unconditional branch instruction to be executed next, [[then]] the instruction reading-amount register is set to "0". In the next instruction cycle, the instruction at address 11 is substituted by no operation (NOP). Then the instruction at a new address 100 is fetched and decoded as a general logic instruction and the instruction reading-amount register is set to "0" for execution of that instruction corresponding to the address 100 in the next instruction cycle. Meanwhile, the next instruction at address 101 of the program memory is pre-fetched ~~pre-read~~ and decoded.

Page 7, amend paragraph [0019] as:

[0019] The program example in Fig. 6 is almost the same as ~~about the same with~~ that in Fig. 5, except that the conditional branch instruction chosen in this case ~~this time~~ is a "CALL" or a "RETURN" instruction at address 12. In the instruction cycle at the PC address 200, the instruction at address 12 is substituted by NOP, and the instruction at a new address 200 is fetched and decoded as a general logic instruction. Thus, the instruction reading-amount register is set to "0" and the instruction at the address 200 will be executed in the next instruction cycle. The instruction at address 201 is pre-fetched ~~pre-read~~ and interpreted as a return instruction, and in the next instruction cycle, the instruction at address 201 is substituted by NOP, then the instruction at the return address 13 is pre-fetched ~~read~~ and decoded.

Page 7, amend paragraph [0020] as:

[0020] According to ~~abovesaid~~ the above description, it is understood that reading two instructions is necessary only when a pre-fetched ~~pre-read~~ instruction is a conditional branch one, otherwise (about 80%) only one instruction has to be pre-fetched ~~is needed to be read~~ so as to avoid ~~save breath of~~ reading unnecessary program memory for reducing power consumption.

AMENDMENTS TO THE DRAWINGS:

FIG. 3 is amended to change "Pre-read" to --Pre-fetch-- in block 302, and "read" to --pre-fetched-- in blocks 303, 304, 305 and 306 as marked in the attached drawings.

FIGs. 5-6 are amended to correct the annotations as marked in the attached drawings.